**Hàng đợi Queue là gì? Cấu trúc dữ liệu và các cách cài đặt Queue**

Trong hướng dẫn này mình sẽ giới thiệu các bạn một cấu trúc lưu trữ dữ liệu nữa đó chính là hàng đợi Queue. Đây là một cấu trúc rất phổ biến trong lập trình và nó rất quen thuộc với chúng ta.



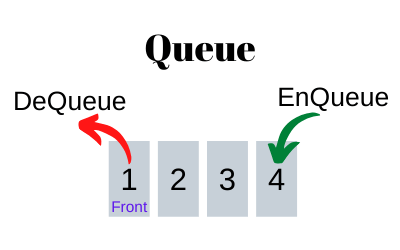
Chúng ta sẽ cùng nhau tìm hiểu về hàng đợi Queue là gì? Các thao tác cơ bản trên hàng đợi Queue cũng như các cách cài đặt nó.

**Mục lục**

* [1. Hàng đợi Queue là gì?](https://freetuts.net/hang-doi-queue-la-gi-3179.html#goto-h2-0)
* [2. Các thao tác cơ bản trong Queue](https://freetuts.net/hang-doi-queue-la-gi-3179.html#goto-h2-1)
* [3. Các cách cài đặt Queue](https://freetuts.net/hang-doi-queue-la-gi-3179.html#goto-h2-2)
* [4. Kết luận](https://freetuts.net/hang-doi-queue-la-gi-3179.html#goto-h2-3)

**1. Hàng đợi Queue là gì?**

Hàng đợi Queue là một cấu trúc trừu tượng. Nó hoạt động dựa trên cơ chế FIFO (first in fisrt out), nghĩa là phần tử nào và trước thì phần tử đó sẽ ra trước.



Ví dụ như lúc nhỏ khi chúng ta xếp hàng vào lớp, thì người đứng ở vị trí đầu tiên là người vào trước, tiếp đến lần lượt các người xếp ở phía sau. Một cấu trúc lưu trữ hoạt động theo cơ chế như vậy được goi là hàng đợi Queue.

*Bài viết này được đăng tại [free tuts .net]*

Để năm rõ hơn về hàng đợi Queue, bây giờ chúng ta sẽ tìm hiểu về các thao tác cơ bản trong hàng đợi nhé.

**2. Các thao tác cơ bản trong Queue**

Về cơ bản thì hàng đợi Queue cũng là một cấu trúc lưu trữ tương tự như Stack, chỉ có điều cơ chế hoạt động của nó ngược lại với Stack, vì vậy trong Queue cũng có các hàm cơ bản tương tư như Stack:

* Hàm khởi tạo Queue rỗng.
* Hàm kiểm tra Queue rỗng.
* Hàm trả về phần tử đầu tiên của Queue.
* Hàm thêm phần tử vào cuối Queue.
* Hàm xóa phần tử đầu tiên của Queue.

Những hàm này chúng ta sẽ được học ở những bài tiếp theo.

Trên đây là một số hàm cơ bản khi làm việc với Queue, ngoài ra còn một số hàm khác phụ thuộc vào bài toán yêu cầu, vì vậy các bạn chỉ cần nắm rõ các hàm này để có thể cài đặt một cấu trúc hàng đợi Queue.

**3. Các cách cài đặt Queue**

Để cài đặt Queue ta cũng sẽ có hai cách cài đặt cơ bản, đó chính là:

* Cài đặt Queue bằng mảng một chiều.
* [Cài đặt Queue bằng danh sách liên kết](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html).

Tuy nhiên khi làm việc với các bài toán, ta ưu tiên chọn cách sử dụng danh sách liên kết để cài đặt hàng đợi, vì trong danh sách liên kết có các ưu điểm như lấy phần tử đầu phần tử cuối, thêm phần tử đầu phần tử cuối.

Nếu chúng ta sử dụng mảng để cài đặt cho hàng đợi Queue, thì khi thực hiện insertList và deleteList tốn rất nhiều thời gian. Trong khi đó nếu sử dụng danh sách liên kết đơn dể cài đặt ta chỉ cần sử dụng con trỏ pHead và pTail là có thể thực hiện nó một cách đơn giản.

**4. Kết luận**

Như vậy là chúng ta đã tìm hiểu xong về hàng đợi Queue là gì? Các thao tác cơ bản trên hàng đợi Queue cũng như các cách có thể cài đặt nó. Ở bài tiếp theo mình sẽ đi chi tiết vào từng cách cài đặt, kèm theo ví dụ cụ thể để các bạn có thể nắm rõ hơn, hãy chú ý theo dõi nhé !!!

# Cài đặt hàng đợi Queue bằng danh sách liên kết

Trong hướng dẫn này mình sẽ giới thiệu các bạn cách cài đặt hàng đợi Queue bằng danh sách liên kết. Đây là một trong hai cách hiệu quả nhất để cài đặt hàng đợi Queue.



Chúng ta sẽ cùng nhau tìm hiểu về cách khởi tạo cấu trúc dữ liệu cho Queue và thực hiện tạo các hàm cơ bản trên hàng đợi. Sau đó sử dụng các hàm đã tạo viết một chương trình nhập xuất đơn giản trong C++.

**Mục lục**

* [1. Cấu trúc dữ liệu của hàng đợi Queue](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h2-0)
* [2. Các hàm cơ bản trong hàng đợi Queue](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h2-1)
  + [Hàm Push](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h3-0)
  + [Hàm Top](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h3-1)
  + [Hàm Pop](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h3-2)
* [3. Ví dụ về hàng đợi Queue](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h2-2)
* [4. Kết luận](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html#goto-h2-3)

## 1. Cấu trúc dữ liệu của hàng đợi Queue

Hàng đợi là một cấu trúc trừu tượng, vì vậy để làm việc với nó ta cần tạo cấu trúc dữ liệu cho nó. Trong hướng dẫn này mình sử dụng danh sách liên kết để tạo Queue vì vậy đầu tiên ta cần có cấu trúc của một Node trong danh sách liên kết.

|  |  |
| --- | --- |
| 1  2  3  4  5 | /\* tạo cấu trúc của Node trong danh sách liên kết \*/  struct Node{      Node \*next;  **int** data;  }; |

Tiếp đến là hai hàm khởi tạo rỗng cho Node và hàm tạo mới một Node, đây là những hàm mà chúng ta đã học ở các bài về danh sách liên kết

*Bài viết này được đăng tại [free tuts .net]*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | /\* khởi tạo rỗng cho Node trong danh sách liên kết \*/  void Init( Queue &q ){      q.head = q.tail = NULL;  }  /\* hàm tạo một Node mới \*/  Node \*createNode( **int** x ){      Node \*p = new Node;      if(!p) **exit**(1);      p->next = NULL;      p->data = x;      return p;  } |

Cuối cùng là tạo một cấu trúc của hàng đợi Queue bằng cách sử dụng Node để tạo, bao gồm hai con trỏ là head và tail để quản lý phần tử đầu và phần tử cuối.

|  |  |
| --- | --- |
| 1  2  3  4 | /\* tạo cấu trúc của hàng đợi Queue \*/  struct Queue{      Node \*head, \*tail;  }; |

## 2. Các hàm cơ bản trong hàng đợi Queue

Trong phần này mình sẽ thực hiện tạo một số hàm cơ bản cần thiết trong hàng đợi Queue: Push(), Top(), Pop().

### Hàm Push

Hàm Push() là hàm để thêm các phần tử vào đầu hàng đợi, đây là một hàm rất quan trọng vì nếu không có dữ liệu trong hàng đợi thì ta không thể thực hiện các thao tác khác.

Việc đầu tiên ta sẽ kiểm tra xem hàng đợi có rỗng hay không, nếu trong Queue không tồn tại phần tử nào thì phần tử được thêm vào chính là pHead và pTail. Ngược lại nếu trong Queue đã tồn tại phần tử thì ta cho pNext của pTail trỏ tới node p.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | /\* hàm thêm phần tử vào đầu trong hàng đợi Queue \*/  void Push(Queue &q, Node \*p ){    //kiểm tra nếu hàng đợi rỗng thì phần tử thêm vào chính là pHead và pTail      if(!q.head) q.head = q.tail = p;    //Nếu trong danh sách đã tồn tại phần tử thì cho pNext của pTail trỏ tới node p    //khi đó node p chính là pTail      else{          q.tail->next = p;          q.tail = p;      }  } |

### Hàm Top

Hàm Top() là hàm để lấy phần tử đầu tiên trong hàng đợi Queue nhưng không xóa nó khỏi Queue. Hay nói cách khác là nó chỉ hiển thị phần tử đầu tiên trong Queue.

Ta thực hiện kiểm tra nếu hàng đợi tồn tại phần tử thì return phần tử đó, ngược lại nếu trong hàng đợi rỗng thì return 0 và kết thúc hàm.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **int** Top(Queue q ){    //kiểm tra nếu hàng đợi tồn tại phần tử thì ta thực hiện return phần tử đầu tiên      if(q.head){          return q.head->data;      }    //Nếu trong hàng đợi rỗng thì return 0 và kết thúc hàm      else return 0;  } |

### Hàm Pop

Hàm Pop() là hàm xóa phần tử đầu tiên trong hàng đợi, đây là một hàm cũng khá quan trọng vì trong một số trường hợp ta cần phải xóa phần tử khỏi Queue.

Việc đầu tiên ta sẽ kiểm tra xem trong Queue có tồn tại phần tử hay không, nếu có tồn tại phần tử thì thực hiện các bước sau:

* Tạo mới một Node p (là Node thay thế cho Node cần xóa).
* Trỏ con trỏ Next đến phần tử tiếp theo để có thể xóa phần tử đầu.
* Gán data cho p và return nó.
* Thực hiện xóa p khỏi hàng đợi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | /\* hàm xóa phần tử đầu trong hàng đợi \*/  **int** Pop(Queue &q ){    //kiểm tra nếu hàng đợi tồn tại thì thực hiện các câu lệnh      if(q.head){      //tạo mới một Node p (là node thay thế cho node cần xóa)          Node \*p = q.head;      //trỏ con trỏ Next đến phần tử tiếp theo để có thể xóa phần tử đầu          q.head = q.head->next;      //gán data cho p và return nó          return p->data;      //sau khi return thì thực hiện xóa nó khỏi hàng đợi          delete p;      }      return 0;// tuong duong queue rong  } |

## 3. Ví dụ về hàng đợi Queue

Trong ví dụ này mình sẽ sử dụng các hàm đã tạo ở trên để thêm một số phần tử và hàng đợi, sau đó tạo một menu với các thao tác như:

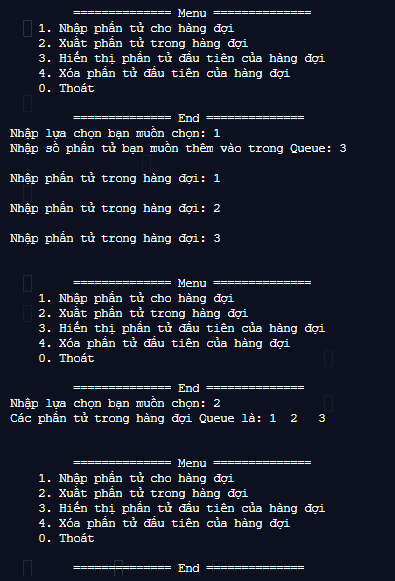
* Thêm phần tử vào Queue.
* Xuất các phần tử trong Queue.
* Lấy phần tử đầu tiên trong Queue.
* Xóa phần tử đầu tiên trong Queue.

Các bạn có thể tham khảo code của mình dưới đây, trong đó mình đã có chú thích rõ ràng.

Full code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121 | /\*\*\*\* Tạo hàng đợi Queue bằng danh sách liên kết đơn \*\*\*\*/  #include<iostream>  using namespace std;  /\* tạo cấu trúc của Node trong danh sách liên kết \*/  struct Node{      Node \*next;  **int** data;  };  /\* tạo cấu trúc của hàng đợi Queue \*/  struct Queue{      Node \*head, \*tail;  };  /\* khởi tạo rỗng cho Node trong danh sách liên kết \*/  void Init( Queue &q ){      q.head = q.tail = NULL;  }  /\* hàm tạo một Node mới \*/  Node \*createNode( **int** x ){      Node \*p = new Node;      if(!p) **exit**(1);      p->next = NULL;      p->data = x;      return p;  }  /\* hàm thêm phần tử vào đầu trong hàng đợi Queue \*/  void Push(Queue &q, Node \*p ){    //kiểm tra nếu hàng đợi rỗng thì phần tử thêm vào chính là pHead và pTail      if(!q.head) q.head = q.tail = p;    //Nếu trong danh sách đã tồn tại phần tử thì cho pNext của pTail trỏ tới node p    //khi đó node p chính là pTail      else{          q.tail->next = p;          q.tail = p;      }  }  /\* hàm lấy phần tử đầu trong hàng đợi, nhưng không xóa nó \*/  **int** Top(Queue q ){    //kiểm tra nếu hàng đợi tồn tại phần tử thì ta thực hiện return phần tử đầu tiên      if(q.head){          return q.head->data;      }    //Nếu trong hàng đợi rỗng thì return 0 và kết thúc hàm      else return 0;  }  /\* hàm xóa phần tử đầu trong hàng đợi \*/  **int** Pop(Queue &q ){    //kiểm tra nếu hàng đợi tồn tại thì thực hiện các câu lệnh      if(q.head){      //tạo mới một Node p (là node thay thế cho node cần xóa)          Node \*p = q.head;      //trỏ con trỏ Next đến phần tử tiếp theo để có thể xóa phần tử đầu          q.head = q.head->next;      //gán data cho p và return nó          return p->data;      //sau khi return thì thực hiện xóa nó khỏi hàng đợi          delete p;      }      return 0;// tuong duong queue rong  }  /\* hàm nhập các phần tử vào hàng đợi \*/  void nhap(Queue &q ){  **int** n, x;    cout<<"Nhập số phần tử bạn muốn thêm vào trong Queue: ";      cin>> n;      while(n--){          cout<<"\nNhập phần tử trong hàng đợi: "; cin>> x;          Node \*p = createNode(x);          Push(q,p);      }  }  /\* hàm xuất các phần tử trong hàng đợi \*/  void xuat(Queue q ){      Node \*p = q.head;      while(p){          cout<<p->data << "\t";          p = p->next;      }      cout<< endl;  }  /\* hàm menu để hiển thị menu và gọi các hàm đã tạo \*/  void menu(Queue q){  **int** lc;    do{          cout << "\n\n\t\t ============== Menu ==============";          cout << "\n\t1. Nhập phần tử cho hàng đợi";          cout << "\n\t2. Xuất phần tử trong hàng đợi";          cout << "\n\t3. Hiển thị phần tử đầu tiên của hàng đợi";          cout << "\n\t4. Xóa phần tử đầu tiên của hàng đợi";          cout << "\n\t0. Thoát";          cout << "\n\n\t\t ============== End ==============";      cout<<"\nNhập lựa chọn bạn muốn chọn: ";          cin>> lc;          switch(lc){              case 0: break;              case 1:                nhap(q);                break;              case 2:                cout<<"Các phần tử trong hàng đợi Queue là: ";                xuat(q);                break;              case 3:                cout<<"Phần tử đầu tiên của hàng đợi là: "<<Top(q);                break;              case 4:                Pop(q);                cout<<"Xóa thành công, vui lòng nhập 2 để kiểm tra lại!!";                break;              default: cout<<"\nNhập sai, vui lòng nhập lại!";          }      } while(lc);  }  /\* hàm main để hiển thị menu\*/  **int** main(){      Queue q;      Init(q);  **int** lc;    menu(q);    cout<<"\n---------------------------------\n";    cout<<"Chương trình này được đang tại Freetuts.net";  } |

**Kết quả:**Mình đã sử dụng thao tác 1 và 2 trong menu, các bạn có thể kiểm tra các thao tác khác nhé.



## 4. Kết luận

Như vậy là chúng ta đã tìm hiểu xong cách cài đặt hằng đợi Queue bằng danh sách liên kết, cũng như tạo các hàm cơ bản trong Queue. Các bạn hãy luyện tập nó thật nhiều để có thể ghi nhớ nhớ thật lâu nhé. Ở bài tiếp theo mình sẽ hướng dẫn các bạn cách cài đặt Queue bằng mảng một chiều, hãy chú ý theo dõi nhé !!

# Cài đặt hàng đợi Queue bằng mảng một chiều

Trong hướng dẫn này mình sẽ giới thiệu các bạn cách cài đằ hàng đợi bằng mảng một chiều. Đây là một trong những cách cơ bản để cài đặt hàng đợi Queue, được áp dụng vào các bài toán đơn giản.



Chúng ta sẽ cùng nhau tìm hiểu về cách cài đặt hàng đợi Queue bằng mảng một chiều như thế nào, sau đó mình sẽ thực hiện một chương trình nhập xuất các giá trị trong Queue.

**Mục lục**

* [1. Các hàm cơ bản trong hàng đợi](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h2-0)
  + [Hàm Push](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h3-0)
  + [Hàm Pop](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h3-1)
  + [Hàm Print](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h3-2)
* [2. Ví dụ xây dựng một cấu trúc Queue](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h2-1)
* [3. Kết luận](https://freetuts.net/cai-dat-hang-doi-queue-bang-mang-mot-chieu-3232.html#goto-h2-2)

## 1. Các hàm cơ bản trong hàng đợi

Trong phần này mình sẽ hướng dẫn viết một số hàm cơ bản trong Queue. Về cơ bản thì khi cài đặt Queue bằng mảng một chiều nó cũng tương tự như cài đăt Queue bằng danh sách liên kết. Chúng ta cũng sẽ cần tạo một số hàm cơ bản như Push, pop và hàm hiển thị các phần tử trong Queue. Ngoài ra còn một số hàm khác tùy thuộc vào bài toán để chúng ta tạo nó, vì như vậy sẽ tối ưu hóa được thời gian viết.

### Hàm Push

Trước khi viết hàm push() để thêm phần tử vào Queue thì ta cần khai báo một mảng queue[] và cấp độ dài cho nó là 100, tùy thuộc vào bài toán chúng ta sẽ khởi tạo độ dài cho phù hợp. Cùng với đó ta cần hai biến khác đó là front và rear, đây là hai biến được khỏi tạo ban đầu là -1. Front là vị trí ban đầu của mảng, rear là vị trí hiện tại mà chúng ta xét đến trong Queue.

*Bài viết này được đăng tại [free tuts .net]*

|  |  |
| --- | --- |
| 1  2 | /\* khởi tạo mảng và các biến cần thiết trong Queue \*/  **int** queue[100], n = 100, front = - 1, rear = - 1; |

Sau khi đã tạo mảng và khởi tạo độ dài cho nó, ta bắt đầu vào viết hàm push().

Đầu tiên ta sẽ tạo một biến val, đây là phần tử ta sẽ thêm nó vào trong Queue. Ta sẽ bắt đầu kiểm tra xem trong phần tử đã đầy hay chưa, nếu Queue đã đầy thì ta không thể thêm phần tử vào được.

Ngược lại nếu trong Queue vẫn còn chỗ trống thì ta sẽ bắt đầu chèn phần tử vào hàng đợi, bằng cách lấy dữ liệu từ bàn phím và gán nó vào rear (vị trị hiện tại trong Queue).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | /\* hàm thêm phần tử vào trong Queue \*/  void push() {  // khởi tạo biến val và đưa nó vào trong hàng đợi  **int** val;  //xét điều kiện hàng đợi đầy, nếu đầy thì không thể thêm phần tử vào được     if (rear == n - 1)     cout<<"Tràn hàng đợi"<<endl;  //Ngược lại nếu trong hàng đợi vẫn còn chỗ trống thì ta thực hiện thêm phần tử     else {        if (front == - 1)        front = 0;        cout<<"Chèn phần tử vào hàng đợi : "<<endl;        //yêu cầu nhập dữ liệu từ bàn phím        cin>>val;        //tăng vị trí hiện tại lên một        rear++;        //thêm phần tử đó vào vị trí hiện tại rear        queue[rear] = val;     }  } |

### Hàm Pop

Trong phần này ta sẽ viết hàm pop() để thực hiện lấy phần tử đầu đồng thời xóa nó khỏi hàng đợi Queue, đây là một thao tác khá quan trọng vì khi thực hiện một số bài toán sẽ yêu cầu xóa phần tử trong Queue.

Cũng như các thao tác khác, ta sẽ kiểm tra xem trong Queue có tồn tại phần tử hay không, nếu Queue rỗng thì ta return và kết thúc hàm. Ngược lại nếu trong Queue tồn tại phần tử ta sẽ thực hiện lấy phần tử đó ra và xóa luôn khỏi Queue.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | /\* hàm xóa phần tử khỏi Queue \*/  void pop() {  //nếu trong hàng đợi rỗng thì không thể xóa phần tử, ta thực hiện return và kết thúc hàm     if (front == - 1 || front > rear) {        cout<<"Không thể xóa phần tử trong hàng đợi ";        return ;     }  //Ngược lại ta sẽ thực hiện lấy phần tử đó ra và xóa luôn khỏi hàng đợi     else {        cout<<"Phần tử đã xóa khỏi hàng đợi là: "<< queue[front] <<endl;        front++;;     }  } |

### Hàm Print

Để có thể kiểm tra và xem được các phần tử trong Queue thì ta cần tạo một hàm đế làm việc đó, cũng tương tự như hàm pop(), ta cần kiểm tra Queue có rỗng hay không, nếu rỗng thì ta return và kết thúc hàm. Ngược lại nếu tồn tại phần tử thì ta sẽ sử dụng vòng lặp for để hiển thị lần lượt các phần tử trong Queue.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | /\* hàm in các phần tử trong Queue ra màn hình \*/  void print() {  //nếu hàng đợi rỗng ta sẽ thông báo     if (front == - 1)     cout<<"Hàng đợi rỗng!!"<<endl;  //Ngược lại nếu hàng đợi có phần tử thì ta sẽ xuất lần lượt các phần tử đó ra màn hình     else {        cout<<"Các phần tử trong hàng đợi là: ";        //sử dụng vòng lặp for để xuất lần lượt các phần tử        for (**int** i = front; i <= rear; i++)        cout<<queue[i]<<" ";           cout<<endl;     }  } |

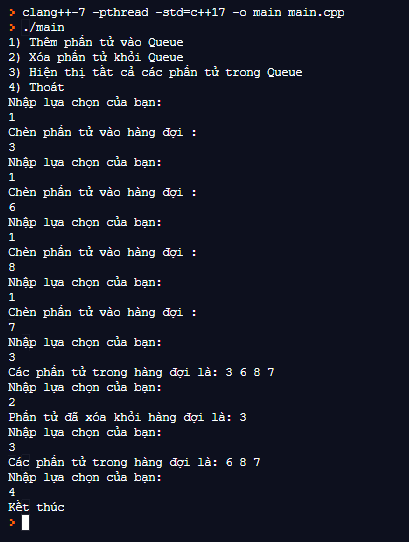
## 2. Ví dụ xây dựng một cấu trúc Queue

Trong phần này mình sẽ sẽ dụng một số hàm đã viết ở trên để thực hiện thêm, xóa, hiển thị phần tử trong Queue, các bạn có thể tham khảo code mình đã viết sẵn dưới đây nhé.

**Full code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | #include <iostream>  using namespace std;  /\* khởi tạo mảng và các biến cần thiết trong Queue \*/  **int** queue[100], n = 100, front = - 1, rear = - 1;  /\* hàm thêm phần tử vào trong Queue \*/  void push() {  // khởi tạo biến val và đưa nó vào trong hàng đợi  **int** val;  //xét điều kiện hàng đợi đầy, nếu đầy thì không thể thêm phần tử vào được     if (rear == n - 1)     cout<<"Tràn hàng đợi"<<endl;  //Ngược lại nếu trong hàng đợi vẫn còn chỗ trống thì ta thực hiện thêm phần tử     else {        if (front == - 1)        front = 0;        cout<<"Chèn phần tử vào hàng đợi : "<<endl;        //yêu cầu nhập dữ liệu từ bàn phím        cin>>val;        //tăng vị trí hiện tại lên một        rear++;        //thêm phần tử đó vào vị trí hiện tại rear        queue[rear] = val;     }  }  /\* hàm xóa phần tử khỏi Queue \*/  void pop() {  //nếu trong hàng đợi rỗng thì không thể xóa phần tử, ta thực hiện return và kết thúc hàm     if (front == - 1 || front > rear) {        cout<<"Không thể xóa phần tử trong hàng đợi ";        return ;     }  //Ngược lại ta sẽ thực hiện lấy phần tử đó ra và xóa luôn khỏi hàng đợi     else {        cout<<"Phần tử đã xóa khỏi hàng đợi là: "<< queue[front] <<endl;        front++;;     }  }  /\* hàm in các phần tử trong Queue ra màn hình \*/  void print() {  //nếu hàng đợi rỗng ta sẽ thông báo     if (front == - 1)     cout<<"Hàng đợi rỗng!!"<<endl;  //Ngược lại nếu hàng đợi có phần tử thì ta sẽ xuất lần lượt các phần tử đó ra màn hình     else {        cout<<"Các phần tử trong hàng đợi là: ";        //sử dụng vòng lặp for để xuất lần lượt các phần tử        for (**int** i = front; i <= rear; i++)        cout<<queue[i]<<" ";           cout<<endl;     }  }  /\* hàm main để tạo menu và gọi các hàm đã tạo ở trên \*/  **int** main() {  **int** ch;     cout<<"1) Thêm phần tử vào Queue"<<endl;     cout<<"2) Xóa phần tử khỏi Queue"<<endl;     cout<<"3) Hiện thị tất cả các phần tử trong Queue"<<endl;     cout<<"4) Thoát"<<endl;     do {        cout<<"Nhập lựa chọn của bạn: "<<endl;        cin>>ch;        switch (ch) {           case 1: push();           break;           case 2: pop();           break;           case 3: print();           break;           case 4: cout<<"Kết thúc"<<endl;           break;           default: cout<<"Lựa chọn của bạn không đúng"<<endl;        }     } while(ch!=4);       cout<<"\n--------------------------\n";     cout<<"Chương trình này được đăng tại Freetuts.net";  } |

**Kết quả:**



## 3. Kết luận

Như vậy là chúng ta đã tìm hiểu xong cách cài đặt Queue bằng mảng một chiều, cũng như ở bài trước ta cũng đã tìm hiểu [cách cài đặt Queue bằng danh sách liên kết.](https://freetuts.net/cai-dat-hang-doi-queue-bang-danh-sach-lien-ket-3180.html) Đây là hai cách cài đặt quan trọng và cơ bản nhất, tuy nhiên khi lập trình ta ưu tiên sử dụng danh sách liên kết để cài đặt, vì các thao tác push và pop sẽ được thực hiện nhanh hơn. Ở bài tiếp theo mình sẽ thực hiện một vài ví dụ sử dụng cả Queue và Stack, hãy chú ý theo dõi nhé!!!

# Tìm các số chẵn lẻ bằng Queue và Stack

Trong hướng dẫn này mình sẽ viết một chương trình sử dụng Queue và Stack để lưu trữ các dữ liệu nhập vào từ bàn phím. Đây là một bài tập áp dụng các hàm cơ bản trong Queue và Stack.



Để làm được bài này các bạn cần có kiến thức về cấu trúc Queue và Stack, các bạn có thể xem lại các bài trước mình đã hướng dẫn.

**Mục lục**

* [1. Đề bài và gợi ý cách thực hiện](https://freetuts.net/tim-cac-so-chan-le-bang-queue-va-stack-3233.html#goto-h2-0)
  + [Đề bài](https://freetuts.net/tim-cac-so-chan-le-bang-queue-va-stack-3233.html#goto-h3-0)
  + [Gợi ý cách thực hiện](https://freetuts.net/tim-cac-so-chan-le-bang-queue-va-stack-3233.html#goto-h3-1)
* [2. Chương trình nhập xuất các số chẵn lẻ trong Queue và Stack](https://freetuts.net/tim-cac-so-chan-le-bang-queue-va-stack-3233.html#goto-h2-1)
* [3. Kết luận](https://freetuts.net/tim-cac-so-chan-le-bang-queue-va-stack-3233.html#goto-h2-2)

## 1. Đề bài và gợi ý cách thực hiện

Trong phần này chúng ta sẽ cùng nhau tìm hiểu để bài toán và gợi ý cách thực hiện bài toán như thế nào nhé.

### Đề bài

Thực hiện một chương trình nhận dữ liệu từ bàn phím. Tạo cấu trúc Queue và Stack sau đó thêm các phần tử là số chẵn vào ngăn xếp Stack và các phần tử số lẻ vào hàng đợi Queue. Hiển thị danh sách ngăn xếp và hàng đợi ra màn hình.

*Bài viết này được đăng tại [free tuts .net]*

### Gợi ý cách thực hiện

Trong bài toán này ta cần hai cấu trúc lưu trữ là Queue và Stack, vậy nên ta sẽ tạo cấu trúc và các hàm cơ bản của nó.

* Cấu trúc dữ liệu của Queue và Stack (danh sách kiên kết hoặc mảng một chiều).
* Hàm Push: thêm phần tử vào Stack.
* Hàm Pop: xóa phần tử khỏi Stack.
* Hàm PushQ: thêm phần tử vào Queue.
* Hàm PopQ: xóa phần tử khỏi Queue.
* Hàm Input: thêm phần tử chẵn vào Stack và phần tử lẻ vào Queue.
* Hàm Print: hiển thị danh sách Queue và Stack ra màn hình.

Trên đây là các hàm cần thiết cho bài toán, chúng ta không nên tạo các hàm không sử dụng trong bài toán, vì như vậy sẽ tốn thời gian và bộ nhớ. Ở các bài trước mình đã hướng dẫn các bạn thực hiện các hàm này rồi, các bạn có thể xem lại hoặc tham khảo code mình sẽ viết dưới đây.

## 2. Chương trình nhập xuất các số chẵn lẻ trong Queue và Stack

Chúng ta sẽ tạo lần lượt các hàm cần thiết theo như gợi ý ở trên, đầu tiên ta sẽ tạo cấu trúc dữ liệu cho Queue và Stack, trong phần này mình sử dụng mảng để cài đặt cho Stack và danh sách liên kết để cài đặt cho Queue.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | /\* khởi tạo biến toàn cục m \*/  #define m 100  /\* khởi tạo cấu trúc Stack sử dụng mảng\*/  struct stack  {  **int** data[m];  **int** top;  };  /\* khởi tạo cấu trúc Queue sử dụng danh sách liên kết \*/  struct Queue  {  **int** Head,Tail;  **int** Data[m];  }; |

Sau khi đã có cấu trúc dữ liệu của Stack, ta sẽ bắt đầu tạo hai hàm cơ bản nhất trong Stack đó chính là hàm Push() và hàm Pop() để thêm và lấy phần tử trong Stack.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | /\* hàm thêm phần tử vào Stack \*/  void Push(stack &a, **int** c)  {      if(a.top==m-1)      {          cout<<"\n Ngăn xếp Stack rỗng!!!";  **exit**(0);      }      else      {          a.data[++a.top]=c;      }  }  /\* hàm xóa phần tử khỏi Stack \*/  **int** Pop(stack &a)  {      if(a.top==-1)      {          cout<<"\n Ngăn xếp Stack rỗng!!!";          return 0;      }      else          return a.data[a.top--];  } |

Tương tư như Stack, ta cũng sẽ tạo hai hàm PushQ() và hàm PopQ() để thêm và xóa các phần tử trong Queue. Kèm theo đó ta sẽ tạo thêm một hàm IsEmpty() để kiểm tra hàng đơi rỗng.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | /\* hàm thêm phần tử vào trong Queue \*/  void PushQ(Queue &q,**int** x)  {  **int** vt;      vt=(q.Tail+1)%m;      if(vt==q.Head)      {          cout<<"\n Hàng đợi đã đầy!!!";  **exit**(1);      }      else      {          q.Data[vt]=x;          q.Tail=vt;      }  }  /\* hàm kiểm tra hàng đợi rỗng \*/  **int** IsEmpty(Queue &q)  {      return (q.Head==q.Tail?1:0);  }  /\* hàm xóa phần tử khỏi hàng đợi \*/  **int** PopQ(Queue &q)  {    **int** vt;  **int** iteam;      while(!IsEmpty(q))      {          vt=(q.Head+1)%m;          iteam=q.Data[vt];          q.Head=vt;          return iteam;      }      return -1;  } |

Và cuối cùng là hàm Input() để thêm các phần tử là số chẵn vào Stack và các phần tử là số lẻ vào Queue. Ta sẽ sử dụng vòng lặp While để lặp hết các phần tử được nhập từ bàn phím, nếu là số chẵn thì ta Push nó vào Stack, nếu là số lẻ thì ta PushQ nó vào Queue.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\* hàm đưa phần tử vào Stack và Queue theo yêu cầu của bài toán     số chẵn vào Stakc và số lẻ vào Queue   \*/  void Input(Queue &q, stack &p)  {  **int** x;      while(1)      {          cout<<"\n Nhập giá trị x, nhập -1 để thoát:";          cin>>x;          if(x==-1)break;          if(x%2==1)          {              PushQ(q,x);          }          else          {              Push(p,x);          }      }  } |

Như vậy là chúng ta đã viết xong hàm Input các phần tử vào hai cấu trúc dữ liệu, bây giờ chỉ cần sử dụng hàm Pop() và PopQ() để lấy các phần tử trong Stack và Queue ra để kiểm tra.

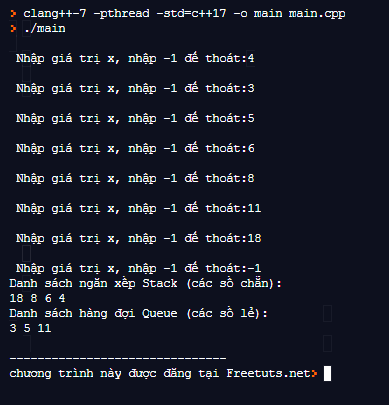
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | /\* hàm in các phần tử trong Queue và Stack \*/  void Print(Queue &q, stack &p){    cout<<"Danh sách ngăn xếp Stack (các số chẵn): "<<"\n";      while(p.top!=-1)      {          cout<<Pop(p)<<" ";      }      cout<<"\n";  **int** t=PopQ(q);      cout<<"Danh sách hàng đợi Queue (các số lẻ): "<<"\n";      while(t!=-1)      {          cout<<t<<" ";          t=PopQ(q);      }          cout<<"\n";  } |

Các bạn có thể tham khảo chương trình hoàn chỉnh mình đã viết dưới đây.

**Full code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127 | #include<iostream>  /\* khởi tạo biến toàn cục m \*/  #define m 100  using namespace std;  /\* khởi tạo cấu trúc Stack sử dụng mảng\*/  struct stack  {  **int** data[m];  **int** top;  };  /\* khởi tạo cấu trúc Queue sử dụng danh sách liên kết \*/  struct Queue  {  **int** Head,Tail;  **int** Data[m];  };  /\* hàm thêm phần tử vào Stack \*/  void Push(stack &a, **int** c)  {      if(a.top==m-1)      {          cout<<"\n Ngăn xếp Stack rỗng!!!";  **exit**(0);      }      else      {          a.data[++a.top]=c;      }  }  /\* hàm xóa phần tử khỏi Stack \*/  **int** Pop(stack &a)  {      if(a.top==-1)      {          cout<<"\n Ngăn xếp Stack rỗng!!!";          return 0;      }      else          return a.data[a.top--];  }  /\* hàm thêm phần tử vào trong Queue \*/  void PushQ(Queue &q,**int** x)  {  **int** vt;      vt=(q.Tail+1)%m;      if(vt==q.Head)      {          cout<<"\n Hàng đợi đã đầy!!!";  **exit**(1);      }      else      {          q.Data[vt]=x;          q.Tail=vt;      }  }  /\* hàm kiểm tra hàng đợi rỗng \*/  **int** IsEmpty(Queue &q)  {      return (q.Head==q.Tail?1:0);  }  /\* hàm xóa phần tử khỏi hàng đợi \*/  **int** PopQ(Queue &q)  {    **int** vt;  **int** iteam;      while(!IsEmpty(q))      {          vt=(q.Head+1)%m;          iteam=q.Data[vt];          q.Head=vt;          return iteam;      }      return -1;  }  /\* hàm đưa phần tử vào Stack và Queue theo yêu cầu của bài toán     số chẵn vào Stakc và số lẻ vào Queue   \*/  void Input(Queue &q, stack &p)  {  **int** x;      while(1)      {          cout<<"\n Nhập giá trị x, nhập -1 để thoát:";          cin>>x;          if(x==-1)break;          if(x%2==1)          {              PushQ(q,x);          }          else          {              Push(p,x);          }      }  }  /\* hàm in các phần tử trong Queue và Stack \*/  void Print(Queue &q, stack &p){    cout<<"Danh sách ngăn xếp Stack (các số chẵn): "<<"\n";      while(p.top!=-1)      {          cout<<Pop(p)<<" ";      }      cout<<"\n";  **int** t=PopQ(q);      cout<<"Danh sách hàng đợi Queue (các số lẻ): "<<"\n";      while(t!=-1)      {          cout<<t<<" ";          t=PopQ(q);      }          cout<<"\n";  }  /\* hàm main để gọi các hàm đã viết \*/  **int** main()  {      stack p;      Queue q;      p.top=-1;      q.Head=q.Tail=0;      Input(q,p);      Print(q, p);      cout<<"\n-------------------------------\n";    cout<<"chương trình này được đăng tại Freetuts.net";  } |

**Kết quả:**



## 3. Kết luận

Như vậy là chúng ta đã thực hiện xong chương trình nhập xuất các số chẵn, lẻ trong Stack và Queue. Các bạn có thể áp dụng bài toán này để luyện tập với các thuật toán khác như số nguyên tố, số chính phương,... . Hãy luyện tập thật nhiều để thành thạo các cấu trúc dữ liệu này nhé, chúc các bạn thực hiện thành công !!!